IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2019, accepted July 19, 2019, date of publication July 24, 2019, date of current version August 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930814

SMAP Fog/Edge: A Secure Mutual Authentication Protocol for Fog/Edge

MAYURESH SUNIL PARDESHI[1] **AND SHYAN-MING YUAN**[2]
[1]Electrical Engineering and Computer Science (EECS-IGP), National Chiao Tung University, Hsinchu 30010, Taiwan
[2]Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan

Corresponding author: Shyan-Ming Yuan (smyuan@cs.nctu.edu.tw)

ABSTRACT Security is a crucial factor for the appropriate functioning of fog/edge computing. Secured mutual authentication in networks has become key demand as per the current security standard. Several applications are in its requirements like wireless sensor network (WSN), Distributed Systems, Micro-Cloud, Smart City, Smart Industry 4.0. Problem statement is ''Design and implementation of Fog servers and edge devices to dynamically interconnect with each other using secured mutual authentication'', which is an NP complete problem. Implementation of secured mutual authentication protocol (SMAP) using techniques of pseudo-random number generator, time-stamps and hash functions can only be considered to evaluate the best performance for connecting large number of smart devices. Our protocol avoids storing master secret keys and repetition of session keys, which makes it more secure and carries no overhead. The experimental results show that the secured mutual authentication system is efficient in comparison to recent benchmarks.

INDEX TERMS Fog layer, edge layer, fog/edge security, security, mutual authentication, micro-cloud, security protocol.

I. INTRODUCTION

There are several mutual authentications schemes within broader scope of information security. Popular standard schemes known are Diffie-Hellman [1] key exchange public key protocol that requires prime and primitive value calculation with shared secret key. Whereas Kerberos [1], a trusted third party authentication protocol requires exclusive ticket granting server and service server which faces single-point failure, stricter time requirements, etc. So deploying traditional authentication schemes requires high-end calculations and large infrastructure which is not portable and unsuitable for resource-constrained devices of the Internet of Things (IoT) [8], [19], [21]. Also, separate management and maintenance of such infrastructure is not cost efficient. Even though the previous schemes may be found efficient on fixed network architecture with high system configuration requirements, still their optimum performance cannot be assured with resource-constrained configuration and dynamic location changing devices. Thus, the need is felt for secured mutual authentication in emerging fog/edge computing paradigm [22].

The associate editor coordinating the review of this manuscript and approving it for publication was Laurence T. Yang.

SMAP Fog/Edge Objectives:

1. Achieve the best performance for resource-constrained devices: As the resource constraint devices have limited system configuration, it can only support less calculation for processing.
2. Authentication of dynamical location changing devices: SMAP does not store any secret key on server and hence is not required to be in a fixed place in the network. It can work in portable mode in the wireless network.
3. Use minimum infrastructure as possible: As SMAP does not require Ticket Granting Server (TGT) and Service Server additionally, it can keep list of public keys on any Fog server node with backup keys on other Fog nodes.
4. Unique keys for all levels and mutual authentication within fog/edge: SMAP uses timestamp for generating a new temporal and final session keys for each operation in fog/edge. Thus, it makes the system stronger than any other protocol.
5. Different security techniques for each level of mutual authentication within fog/edge: SMAP uses three different types of pseudo-random number generators with

the best configurations for its effectiveness. Hence, each level differs in its features of key generation.

Several applications encourage the development of Fog/Edge ahead, some of them are stated as below [24]:

A. SMART UNIVERSITY'S

Presence of multiple Fog servers [11] deployed within a university environment, where laboratory servers are utilized. The Fog server in a different laboratory having specialized facility's that include GPU processing, CUDA processing, high-performance parallel servers which can be utilized for high-end image processing and scientific job processing shared by various laptops, mobile devices, sensor's, etc. Underutilized systems will, therefore, be properly used by resource scare/demanding systems. Synchronizing with the respective departments will provide several event information to the students, faculty and staff.

B. SMART LABORATORY'S

A micro-cloud [9], [11] can be created within a laboratory or a meeting room by treating some routers, desktop systems as fog servers. These systems may or may not be connected to the internet but can do processing of jobs in the distributed processing systems approach. Thus, underutilized systems will be effectively used within a micro-cloud network. Simulations, resource sharing, high performance distributed processing and parallel processing will be some of its applications.

C. SMART INDUSTRY

Smart Industry 4.0 [3], [6], [9] is a current trend in large industries having sensors at the fog/edge. Several fog servers are present at the administrative offices that collect data from groups of sensors for temperature monitoring, gas leakages, smart meters, water sprayers, etc. are used within nuclear power plants. Edge layer devices process the data at the Fog layer, generated by the sensor and can be used to send control commands to the actuator control on system maintenance, thus keeping humans out of danger [25].

D. SMART VEHICLES

Popularly known as connected vehicles [3], [25] which maintains active connection between several cars, bikes, bicycles, etc. for showing appropriate routes, traffic jams indications. Smart cars have inbuilt processor's and storage that allows to process routes and can co-operate with other vehicles for rally, reaching a target, accidental cases reporting to nearby hospitals, emergency units. These sensors maintain active monitoring, storing records of test cases, predicting nearby gas stations, storing data for next journey, self-driving cars are some of its applications. The high processing power available with many vehicles allows exploring several possibilities and providing best in class facilities.

The organization of our paper is given as below. Section II. Literature Survey discuses features of recent security in fog/edge computing models. Section III. Methodology presents our SMAP protocol functioning and operations with various security attacks prevention is proved by supporting reasons. Section IV. Results and Discussion gives achieved results with SMAP protocol and comparison with benchmarks. Section V. Conclusion gives the objectives mapping and outcome, followed by references and acknowledgment.

II. LITERATURE SURVEY

As the current systems are developing for adapting to fog/edge architecture, the security and privacy aspect is not progressed enough to predict the requirements for its safety [6], [9], [11]. Even though closely matching the architecture of machine to machine (M2M) [15] and smart grids were studied enough. Hence, we will see some literature survey that somehow associates with concerns of the fog:

Mobile devices are in large numbers as compared to any other, at present. Identity-based authentication solutions are provided using a lightweight equipment certification program, having efficiency improvement and reduced bandwidth consumption for transmission. The proposed work by Deng and Li [7] considers: *(1) Key management flexibility by avoiding secure channel to get keys, (2) Public key system decreases authentication and key negotiation burden and thus improving efficiency (3) Ease of authentication and digital signature's implementation in a distributed environment.* Here, the absence of secure channel for key exchange, a public key model of large dependency and high digital signature calculations are unsuitable for massive fog node based architecture.

A recent survey presented by Mukherjee *et al.* [6], shares various insights about the current fog/edge security concerns and challenges. *(1) How to safeguard data against malicious or malfunctioning fog nodes attacks, (2) How to identify malicious insider and exclude him from the system as he steals end users private key and illegitimately access data, (3) How the EU is able to mutually authenticate new fog/edge user within a network to connect with fog server securely, (4) How to support location and identity privacy in UAV assisted fog computing and (5) How to achieve client-to-server and server-to-client authentication* represents a challenge as conveyed.

A detailed survey of optimization on IoT public key infrastructure by Kelly and Hammoudeh [5] represents how IoT faces unique security challenges in resource-constrained environment. *As IoT is limited by computing, storage, memory and power, it's hard to get deployed of advanced security protocol, which acts as an overhead*. PKI is considered again as a secure environment for such devices. Recent research only shows attempts to reduce the size of X.509 certificates or orchestration of DTLS handshake. Virtual resources are also considered that perform implementation of the complex cryptographic protocols as a substitute but still face high dependency and computing challenges. Fragmentation of encryption handshake can be avoided by the use of efficient DTLS handshake protocol frame fit into 802.15.4 packets.

Use of symmetric key for overall communication is still possible threat to all data security in this system.

One Time Password with ECC, a two-factor authentication scheme is approached by Shivraj *et al.* [4]. The scheme presented here is a hybrid system combining Lamport's One Time Password (OTP) and Identity Based Elliptical Curve Cryptography (IBE-ECC). This scheme is tested with the current authentication system and approached for real-time IoT networks and smart cities. Having a smaller key size and less infrastructure is the objective for showing the optimal performance of that system. Even though the system performs well in comparison by proof with other OTP algorithms in IoT, we will be having genuine algorithms for the new fog system architecture.

Octopus, a mutual authentication scheme is presented by Ibrahim [3]. In this scheme, a new user randomly roaming in the system can mutually authenticate him using a master secret key and also the keys are shared with multiple servers joining the network. The scheme is designed for smart card/devices. As the scheme is better utilized in smart cards and devices its mostly in a fixed manner and hence may face the possibility of masquerading servers, users for re-authentication and repetition of master password with symmetric key encryption/decryption. Nevertheless, repeating a master password is still considered to be unsafe in large scale fog/edge network model.

Computational complexity on combinatorial problems presented by karp [26], presents a problem on making a maximum number of requests that can be made simultaneously within a network using the concept of Edge-Disjoint Paths (EDP) or Computing Maximum Flow is proved to be NP-Complete. Ultimately, we are trying to overcome this problem by using cryptographic techniques and mathematical functions that can give high security with fewer calculations as possible, to achieve the large number of mutual authentication within a network containing millions of devices with less possible time.

Lightweight mutual authentication protocol for IoT and its applications is proposed by Li *et al.* [28]. Lightweight protocol used between two lightweight devices operates on public key encryption scheme. This protocol uses challenge and response scheme by encryption for mutual authentication. This type of scheme is represented as an n-pass protocol, as the security level and parameters for encryption determine the number of rounds. The protocol assumes the participants are already aware of their respective public keys and identities. The protocol is found to be performing well when compared to RSA, ECC and their original protocol version as compared to their own optimal scheme.

Securing IoT smart homes by access control and mutual authentication is presented by Alshahrani and Traore [32]. An updatable temporary identity and session key for every session by an anonymous and lightweight key exchange and mutual authentication is proposed. Focusing on specific nodes communication, a virtual domain segregation is used for division of devices by a security policy in IoT smart homes. Hence, insider and outsider threats are made less severe by using such lightweight methods. Identity thefts are avoided using by using a new fog computing architecture in IoT devices. This methodology is then compared with other recent benchmarks for communication cost number of messages and total number of bits.

TABLE 1. Notations.

Notation	Meaning
c	Client / Edge User
s	Server / Fog Server
AS	Authentication Server
a_x	Network IP address of x
v	Validity time of message
t_x	Timestamp of x
PK_x	Public key of x
SK_x	Private key of x
R_x	Random number of x
M	Message
$H(x)$	A hash SHA-256 bit function with input x
I_x	Identity of x
E_x	Encryption by x's public key
D_x	Decryption by x's private key
$K_{x,y}$	Session key from x to y
$AS_{x,y}$	Authentication from x to y
$X \rightarrow Y$	X computes and sends to Y

III. METHODOLOGY

The operation for mutual authentication involves at least two communicating parties *c, s* looking forward establishing secure communication and an authentication server *AS* to initiate the process.

The protocol ensures confidentiality by the use of public key encryption, integrity by one-way hash function and authentication by mutual agreement between the communicating parties for deciding a final session key in an event of communication. Identity of the system is useful for checking the accuracy of sent message I_x, whereas time-stamp t ensures that the message operated is current. All the public keys are stored in the authentication server *AS*, which helps in registration and authentication of the system.

In SMAP protocol, we store the public key PK_x of all the systems including fog/edge server's and client's in Authentication Server *AS*.As the fog/edge system is dynamic, we always keep a backup *AS*, in case the original server leaves the network. Hence, *AS* stores all client and server's public keys PK_c, PK_s and is responsible for initiating mutual authentication within the network. For every client/server mutual authentication based protocol needs to be complete, to establish trust and initiate secured communication.

A. SECURE MUTUAL AUTHENTICATION PROTOCOL (SMAP) PROCESS

Referring to figure 1, the SMAP protocol can be explained as follows with reference to the notation table 1 [1], [12]:

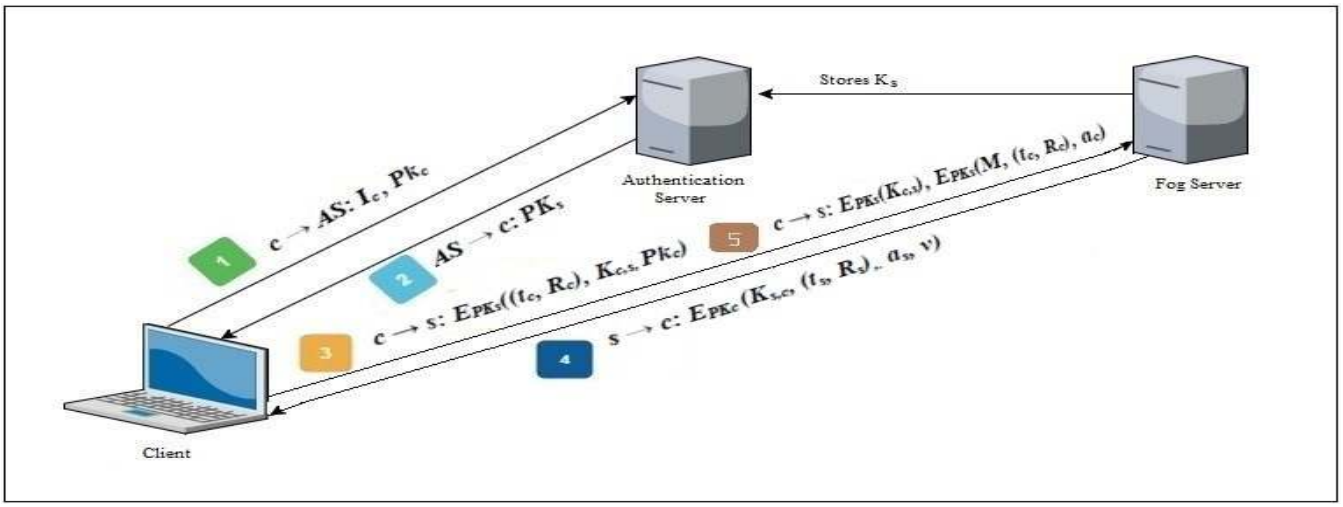

FIGURE 1. SMAP protocol.

1. Every client requiring to establish secure connection with a fog server, requests for its public key from *AS*.The client sends its own identity I_c and public key Pk_c for its validation.

$$\boldsymbol{c \rightarrow AS{:}I_c, \quad Pk_c}$$

2. The *AS* confirms client c as a valid user and replies to request by a public key of requested fog server $\boldsymbol{PK_s}$.

$$\boldsymbol{AS \rightarrow c{:}PK_s}$$

3. The client c then generates a pseudo-random number R_c and xored with the timestamp t_c. Client c then calculates its session key by $\boldsymbol{K_{c,s} = H(t_c \oplus R_c)}$. All these parameters are then grouped and encrypted by the fog server's public key $\boldsymbol{E_{PKs}}$ sent by *AS*for input to Lightweight Encryption Scheme (LES) [28]. Note here, t_c, R_c and Pk_c are sent for calculation of $k_{c,s}$ and confirming its identity respectively.

$$\boldsymbol{c \rightarrow s{:}E_{PKs}((t_c, R_c), K_{c,s}, Pk_c)}$$

4. After receiving the encrypted message from client c, the fog server decrypts it by using his private key $\boldsymbol{D_{SKs}}$ and confirms the calculation of $K_{c,s}$ by using a hash of t_c *xor* R_c. In a similar way, the fog server generates its session key $\boldsymbol{K_{s,c} = H(t_s \oplus R_s)}$. Also, the fog server's absolute network IP address a_s and validity of message v are used. All these parameters are then encrypted using the client's public key $\boldsymbol{E_{PKc}}$ and sent to the client.

$$\boldsymbol{s \rightarrow c{:}E_{PKc}(K_{s,c}, (t_s, R_s)., a_s, v)}$$

5. Client c decrypts the received message by using his private key $\boldsymbol{D_{SKc}}$ and for confirmation re-calculates session key $K_{s,c}$ by using a hash of t_s *xor* R_s, stores client's network address a_s and timestamp validation of message made by v. Ultimately, the client re-calculates its session key by $\boldsymbol{K_{c,s} = H(t_c \oplus R_c)}$. In the first part of the message, new session key $K_{c,s}$ is encrypted by using the fog server's public key $\boldsymbol{E_{PKs}}$. Whereas, the second part of message, is again encrypted with fog server's public key $\boldsymbol{E_{PKs}}$ containing plain-text message M grouped with client's timestamp t_c, pseudo-random number R_c and network address a_c which is later confirmed by the fog server by calculating hash of $\boldsymbol{H(t_c \oplus R_c)}$ with client's session key $k_{c,s}$. Both this key and message encryption are attached and sent as single message to fog server.

$$\boldsymbol{c \rightarrow s{:}\ E_{PKs}(K_{c,s}), \quad E_{PKs}(M, (t_c, R_c), a_c)}$$

Note: The pseudo-random random number generator used in step 3, 4 and 5 are each with different category of PRNG generators. Also, The LES algorithm takes different random values in its algorithm as input, hence the encryption is not always with the same parameters [28].

B. ANALYSIS OF HARDNESS OF PSEUDO-RANDOM NUMBER GENERATOR [2] [4]

We state about the analysis of hardness of our *PRNG* generation with respect to the following situations. Let the c_1 and s_1 are any two user's c or s and t_1 and t_2 be two random time instances.

Lemma 1: Computing a new *PRNG* independently from the same *c or s* and parameters.

Proof: Let $PRNG_{t1}$ and $PRNG_{t2}$, the *PRNGs* generated at two different instances of time *t1* and *t2* respectively for c_1, which are given as

$$PRNG_{t1} = [t1]c_1 \tag{1}$$
$$PRNG_{t2} = [t2]c_1 \tag{2}$$

From the independent $PRNGs$ $PRNG_{t1}$ and $PRNG_{t2}$, computing the *PRNG*, $PRNG_{t3}$ for the c_1 at t_3 will be given as $t_3 > t_2 > t_1$.

Lemma 2: Computing a new *PRNG* for a c *or s* is again independent from other c *or s* at the same instance of time.

Proof: Let $PRNG_t^{C1}$ and $PRNG_t^{S1}$, the *PRNGs* generated at same instances of time on different Fog/Edge layers respectively for c_1 and s_1, which are given as

$$PRNG_t^{C1} = [t1]c_1 \quad (3)$$

$$PRNG_t^{S1} = [t2]s_1 \quad (4)$$

Therefore, computing $PRNG_t^{S1}$ is completely independent from the given $PRNG_t^{C1}$ without knowing $[t2]s_1$ and in actual it is applicable for more than two devices and *PRNGs*

Lemma 3: Computing a new *PRNG* for a *c or s* is independent of another *c or s* at the different instance of time.

Proof: Let $PRNG_t^{C1}$ and $PRNG_t^{S1}$, the *PRNGs* generated at different instances of time t_1 and t_2 on different Fog/Edge layers respectively for c_1 and s_1, which are given as

$$PRNG_t^{C1} = [t1]c_1 \quad (5)$$

$$PRNG_t^{S1} = [t2]s_1 \quad (6)$$

Thus, computing $PRNG_t^{S1}$ is independent from $PRNG_t^{C1}$

Lemma 4: Computing a new PRNG for a *c or s* from the known *PRNG* of same/another Fog layer c*or s* at same/different instance of time is independent from each other

Proof: Referring to *Lemma's* 1-4, this *lemma* is also proved. So here different fog layers are independent of generating *PRNG* with same/different instance of time. Thus, *PRNG* generation in SMAP*PRNG* is stronger.

The Table 3, shows how SMAP can prevent various security attacks [10], [13].

IV. RESULTS AND DISCUSSION

The primary goal of this paper is to analyze both protocol mathematically and experimentally [4], the efficiency of mutual authentication based on pseudo-random number generator *PRNG,*hash and key exchange for Fog/Edge. As a part of this, we have implemented and experimented various *PRNG* compiler parameters including GCC, Borland C/C++ and Turbo Pascal with SHA-256 Hash bit [31] and time stamp having a unique epoch value at every instance of time.

A. SYSTEM CONFIGURATION

For experimentation, we have deployed *PRNG* schemes on Server and Desktop. The *PRNG* generation configuration is deployed on a system, whose configuration is mentioned in Table 2. We have conducted load testing with various compilers on the system with varying number of standard parameters for best performance. To enable this, we utilized python version 2.7.5 [16] on CentOS 7 server platform [14].

We have conducted experiments of generating *PRNG* using Linear Congruential Generator (LCG) and have also analyzed the performance of Mersenne Twister. Both are well-known *PRNG* [2], which follows current standards for generating true *PRNGs*. We have set parameters differently during the calculation of each key to avoid repetition possibilities of the same pseudo-random number generation.

TABLE 2. Experimental setup.

Computing Environment:	Server, Desktop
Hardware:	Intel Core i5-3450U CPU @ 3.10 GHz
Primary Memory:	16 GB
Operating System:	Ubuntu 16.04 LTS, 64-bit
Library (Python):	Time, HashLib, Python WSGI(Web Service Gateway) and Openssl

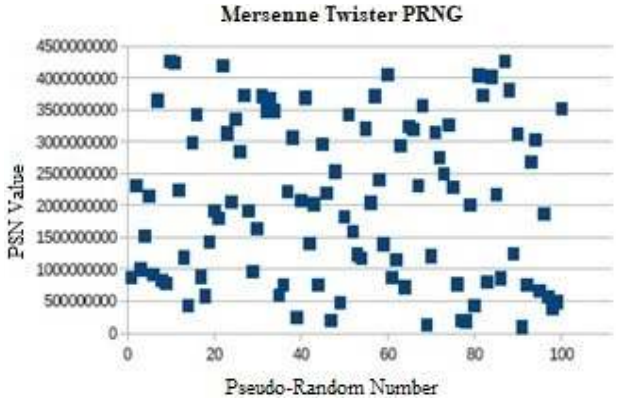

FIGURE 2. Mersenne twister pseudo random number generator.

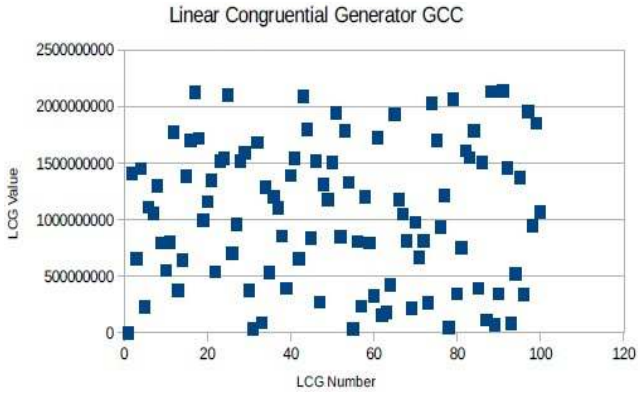

FIGURE 3. Linear congruential generator PRNG of GCC compiler.

B. PERFORMANCE ANALYSIS

In Figure 2, we have analyzed the performance of Mersenne Twister *PRNG* for generating 100 pseudo-random numbers.In Mersenne Twister Pseudo Random Number Generator provides the distribution of 100 values in 10^9, which is a very high range. As it is best known, we have also compared it with its competitor Linear Congruential Generator.

In Figures 3-5, we have shown the generation of 100 LCG pseudo-random numbers with different compiler settings. These settings are unique depending on the parameters of

TABLE 3. SMAP security.

Category	Attack Type	SMAP Prevention
Active Attack	Spoofing	A malicious intruder representing his identity will be unable to authenticate himself for requesting public key of server it wants to communicate. Spoofing will be unable to gain access from authentication server as $c \rightarrow AS : I_c , PK_c$
	Modification	In Fog/Edge wireless system, it can easily detect available systems within the proximity and can communicate with known authentication server, as the operations will not be initiated until a session key $K_{c,s}$ is mutually agreed by both the parties by completing the protocol.
	Wormhole	As the packets are rerouted to the malicious system, it cannot be directly associated with it because of the authentication system. Rerouted packets cannot initiate the protocol process until they receive public key of proximity server. $AS \rightarrow c : PK_s$
	Fabrication	A malicious user may represent false routing of packets. Succeeding in doing it, he still cannot decrypt the messages exchanged in the protocol process as they are secured by a public key of fog server PK_s and later temporal session key for the mutual agreement $k_{s,c}$ and $k_{c,s}$.
	Sinkhole	Selective modification is not possible in SMAP as a hash is calculated for each session keys $K_{c,s}=H(t_c \oplus R_c)$ and $K_{s,c}=H(t_s \oplus R_s)$. Also, the use of timestamp t_s, t_c and message validity v makes it harder to be used in forwarding.
Passive Attack	Traffic Analysis	An attacker monitors the communication path and the amount of data traveling between multiple parties. So the possibility is the attacker may guess the entities within the network but cannot determine operations carried out in each message secured by public key PK_s and session keys $K_{s,c}$ and $K_{c,s}$
	Eavesdropping (Man-in-the-middle)	Eavesdropper captures packets transmitted in the network and tries to read it. This type of attack is succeeded only when the messages lack encryption of data, so it not issue as SMAP encrypts all messages using the public key PK_s and session keys $K_{s,c}$ and $K_{c,s}$. Also, SMAP does not communicate private key SK_c and SK_s. Even if the eavesdropper succeeded in breaking one of the public key or temporal session key, he will not be able to generate new message as it needs valid pseudo-random number generator R_x and timestamp T_x.
	Monitoring	Monitoring can sense network usage and reads the communication of messages between various systems. As long as data is secured by using hashing H, SHA-256 bit and strong cryptographic algorithms LES, passive monitoring is not effective.
Advance Attack	Blackhole	If the router is compromised in between for dis-functioning of the network flow, then the protocol is restarted. In case, the protocol message is held for a long duration of time then the timestamp t_s, t_c and message validity v will eventually lead to termination of the protocol.
	Replay	As the attacker intercepts the packet and resend it repeatedly to create an illusion to the receiver about the original message with some modification, this process would be considered to be invalid. As the protocol needs to complete the process for generating final session key $K_{c,s}=H(t_c \oplus R_c)$, replay attack would not be succeeded in any case without having valid hash H, pseudo-random number R_x and valid timestamp t_c and t_s.
	Rushing	Rushing attack uses a modification of messages with replay. Modification is not possible because of hash function H used and replay is avoided as discussed before.
	Byzantine	A replay loop and non-optimal path for routing will lead to the expiry of timestamp t_s, t_c and invalidation by v. In this case, the protocol will be incomplete and hence terminate. Thus, identification of malicious node and attack would be discovered immediately.
	Location Disclosure	The network identity a_c, a_s of communicating parties and their data traffic are recorded for a possible future attack. In this case, SMAP server is not fixed, it can be dynamic and also no encrypted data by public key PK_s and session keys $K_{s,c}$, $K_{c,s}$. are repeated because of their dynamic factor.

their respective compilers for best performance. The below table 4 shows its detail values.

In Table 4, we can analyze the performance of each LCG compiler having different set of parameters [17]. The time required for generating 100 keys with the portable system is still quite impressive. So generating millions of keys on standard configuration server can be done in hardly one minute of a time.

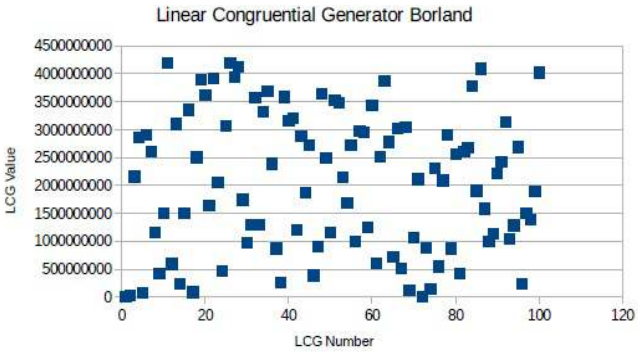

FIGURE 4. Linear congruential generator PRNG of borland C/C++ compiler.

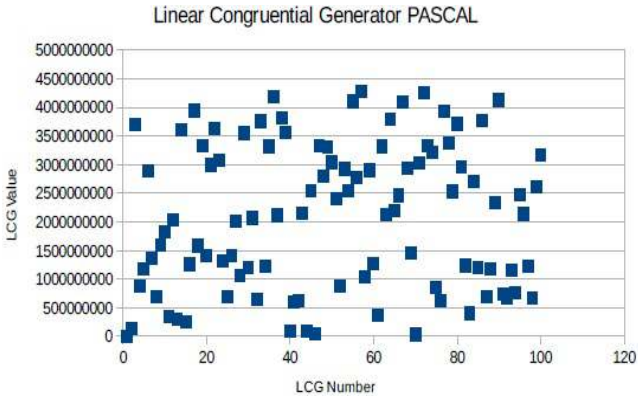

FIGURE 5. Linear congruential generator PRNG of turbo pascal compiler.

TABLE 4. LCG compiler performance analysis.

LCG Compiler	Modulus M	Multiplier A	Increment C	100 keys generation (time in sec)
gcc/glibc	2^{31}	1103515245	12345	2.28e-05
Borland C/C++	2^{32}	22695477	1	2.69e-05
Turbo Pascal	2^{32}	134775813	1	2.59e-05

Generating one million keys is enough at time required for setting all devices in connection in the university campus environment. Whereas, for micro-cloud, there will a bunch of devices easily connected in milliseconds.

In above Figures 6-8, we can see the generation of session keys S_{K1}, S_{K2} and S_{K3}. There are 100 session keys generated in each session as shown in figures based on SHA-256 bit algorithm which takes H(Timestamp $\oplus$ LCG).

In Figure 9, we can see that the time comparison of session keys generation is compared with each other for S_{K1}, S_{K2} and S_{K3} No major difference is being noticed while comparing them. Hence, the performance is quite similar while generating keys from a different set of LCG compiler and parameter values.

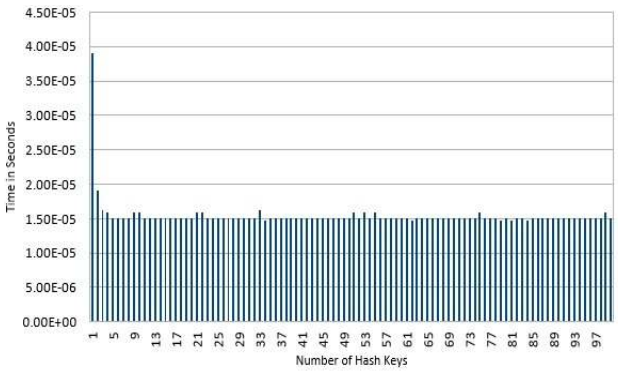

FIGURE 6. Session keys S_{K1} generation time analysis.

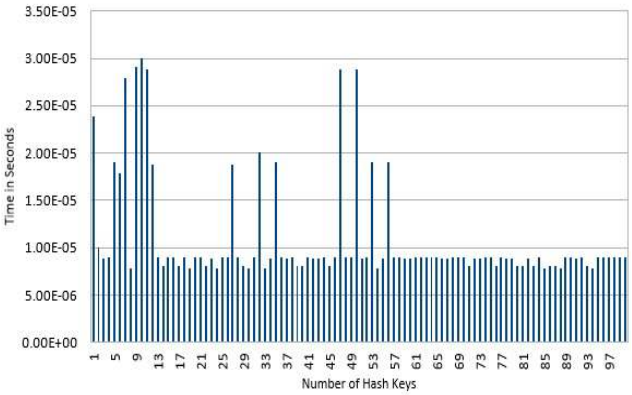

FIGURE 7. Session keys S_{K2} generation time analysis.

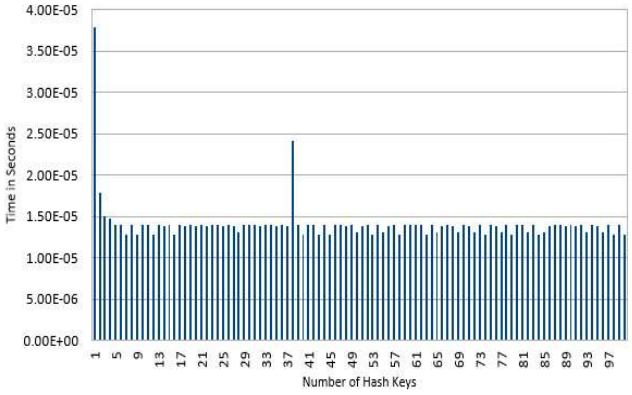

FIGURE 8. Session keys S_{K3} generation time analysis.

The time taken for each session key set is as given in Table 5. The encryption in the session key S_{K3} can be performed using a short public key generation system of Elliptical Curve Cryptography (ECC) [18] can also be best suited for this protocol. Though the time analysis in S_{K3} will differ based on the configuration of the server and broadband connection. The key exchange process of Elliptic Curve Diffie-Hellman(ECDH) [26] is also the best-suited example.

Table 6. shows a comparison of protocol performance of 3MPAKE [27] with SMAP Fog/Edge. The performance time is shown for calculation of 48 sessions including multiple cryptographic encryption and decryption by LES [28] at server and client side respectively.

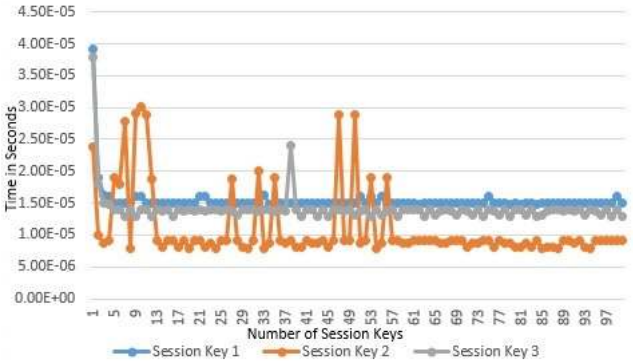

FIGURE 9. Comparison of session keys generation time.

TABLE 5. Session key generation time analysis.

SHA 256-bit Hash	Keys Generation - 100 each (Time in sec)
Session Key S_{K1}	0.001182
Session Key S_{K2}	0.001075
Session Key S_{K3}	0.001137

TABLE 6. Comparison of protocol performance.

Protocol	3MPAKE	SMAP
Computation Time (s)	9.29	5.97

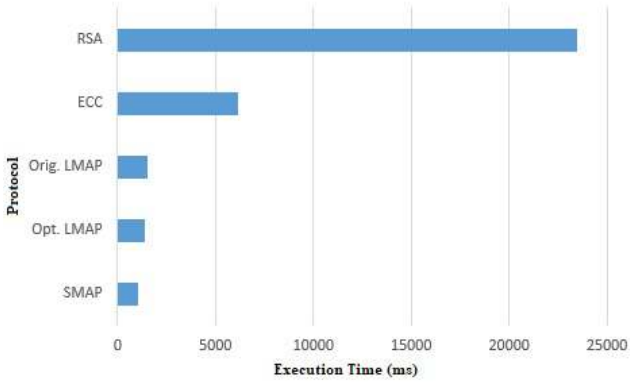

FIGURE 10. Comparison of RSA, ECC and LMAP with SMAP.

The system environment used is Cooja emulator, same configuration for performance evaluation of all in Figure 10. by comparing RSA [29], [31], ECC [29], original LMAP [28] and optimal LMAP [28] all of the security level of 80 bit with SMAP. We have used Contiki OS – Cooja simulator [30] with Mote Type - Tmote Sky@3.9MHz with TX range 50 m for mutual authentication within multiple nodes. The ECC protocol performs handshake by ECDSA. As we can see above, SMAP requires much less time for complete mutual authentication process as compared to other protocols, as SMAP with 1,043ms compared to optimal LMAP 1,401ms, original LMAP 1,535ms, ECC 6,160ms and RSA 23,500ms.

V. CONCLUSION

In this paper, we have achieved the objectives as proven by experimentation of performing less possible operations on resource-constrained device, authenticating dynamic location changing fog/edge devices, use minimum infrastructure as possible, unique keys for all levels and use of different PRNG generators at each session. The novel scheme is designed as per the requirements of the new architecture model. The SMAP model is suitable for defending against various attacks and the result section shows better performance on a large number of devices to be authenticated within a short amount of time. SMAP protocol is best suited for Fog/Edge architecture including IoT that can be utilized in Smart University, Smart Labs, Industry 4.0 and Smart Vehicles, where the secured environment is a high priority. Ultimately, we have achieved an efficient protocol as compared with benchmarks including 3MPAKE, RSA, ECC, LMAP and Optimal LMAP. As a future part, we will perform the experimentation on multiple high-end servers from multiple universities for more availability of resources. Also, backup *AS* is provided in the case to avoid single point failure like kerberos.

ACKNOWLEDGMENT

The authors would like to thank our reviewer's for giving useful corrections to improve our paper and work further. They are grateful for the comments on the previous version of our manuscript and it was our pleasure to revise it. Also thanks to Library and Distributed Computing System (DCS) Laboratory, for allowing to use their resources at NCTU, Taiwan.

REFERENCES

[1] B. Schneier, *Applied Cryptography*, 2nd ed. Hoboken, NJ, USA: Wiley, 1996.
[2] C. Easttom, *Modern Cryptography: Applied Mathematics for Encryption and Information Security*. New York, NY, USA: McGraw-Hill, 2016.
[3] M. H. Ibrahim, "Octopus: An edge-fog mutual authentication scheme," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1089–1101, Nov. 2016.
[4] V. L. Shivraj, M. A. Rajan, M. Singh, and P. Balamuralidhar, "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)," in *Proc. 5th IEEE Nat. Symp. Inf. Technol., Towards Smart World*, Feb. 2015, pp. 1–6.
[5] D. Kelly and M. Hammoudeh, "Optimisation of the public key encryption infrastructure for the Internet of Things," in *Proc. Int. Conf. Future Netw. Distrib. Syst. (ICFNDS)*, Amman, Jordan, Jun. 2018, Art. no. 45.
[6] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.
[7] J. Deng and J. Li, "An identity authentication solution based on ECC for mobile terminal," in *Proc. Int. Symp. Comput. Inform. (ISCI)*, 2015.
[8] G. Zhao, X. Si, J. Wang, X. Long, and T. Hu, "A novel mutual authentication scheme for Internet of Things," in *Proc. Int. Conf. Modelling, Identificat. Control*, Shanghai, China, Jun. 2011, pp. 563–566.
[9] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey," 2018, *arXiv:1810.00305*. [Online]. Available: https://arxiv.org/abs/1810.00305
[10] S. Khan, S. Parkinson, and Y. Qin, "Fog computing security: A review of current applications and security solutions," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 6, no. 1, p. 19, 2017.
[11] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[12] S. Briais and U. Nestmann, "A formal semantics for protocol narrations," in *Proc. 1st Int. Symp. Trustworthy Global Comput. (TGC)*, Edinburgh, U.K., Apr. 2005, pp. 163–181.

[13] M. V. Pawar and J. Anuradha, "Network security and types of attacks in network," *Procedia Comput. Sci.*, vol. 48, pp. 503–506, 2015.
[14] *CentOS 7*. Accessed: Nov. 2018. [Online]. Available: https://www.centos.org/download/
[15] *M2M*. Accessed: Nov. 2018. [Online]. Available: https://www.etsi.org/technologies/internet-of-things
[16] *Python*. Accessed: Nov. 2018. [Online]. Available: https://www.python.org/downloads/
[17] K. Entacher, "A collection of selected pseudorandom number generators with linear structures," ACPC, Vienna, Austria, Tech. Rep., Aug. 1997.
[18] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.
[19] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 169–186.
[20] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," NIST, Gaithersburg, MD, USA, Tech. Rep., Jun. 2011.
[21] G. Gan, Z. Lu, and J. Jiang, "Internet of Things security analysis," in *Proc. ITAP*, Wuhan, China, Aug. 2011, pp. 1–4.
[22] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," *Internet Everything*, pp. 103–130, Oct. 2017.
[23] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, and J. K. Zao, "Fog as a service technology," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 95–101, Nov. 2018.
[24] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 115–124.
[25] A. Kattepur, H. K. Rath, A. Simha, and A. Mukherjee, "Distributed optimization in multi-agent robotics for industry 4.0 warehouses," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.*, Pau, France, Apr. 2018, pp. 808–815.
[26] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.
[27] L. Wenmin, W. Qiaoyan, S. Qi, Z. Hua, and J. Zhengping, "Password-authenticated multiple key exchange protocol for mobile applications," *China Commun.*, vol. 9, no. 1, pp. 64–72, Jan. 2012.
[28] N. Li, D. Liu, and S. Nepal, "Lightweight mutual authentication for IoT and its applications," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 4, pp. 359–370, Oct./Dec. 2017.
[29] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proc. 4th ACM Workshop Secure Ad Hoc Sensor Netw.*, 2006, pp. 169–176.
[30] *Contiki OS*. Accessed: Mar. 2019. [Online]. Available: http://www.contiki-os.org
[31] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Proc. Int. Workshop Fast Softw. Encryption*, in Lecture Notes in Computer Science, vol. 3017. Berlin, Germany: Springer, 2004.
[32] M. Alshahrani and I. Traore, "Secure mutual authentication and automated access control for IoT smart home using cumulative keyed-hash chain," *J. Inf. Secur. Appl.*, vol. 45, pp. 156–175, Apr. 2019.

MAYURESH SUNIL PARDESHI received the Bachelor of Engineering degree in information technology from Mumbai University, in 2010, and the Master of Technology degree in computer science and engineering from the Walchand College of Engineering, Sangli, India, in 2013. He is currently pursuing the Ph.D. degree with the Electrical Engineering and Computer Science (EECS-IGP), National Chiao Tung University, Hsinchu, Taiwan. His current research interests include optimization in grid/cloud computing, security, machine learning, and fog/edge computing.

SHYAN-MING YUAN received the Ph.D. degree in computer science from the University of Maryland at College Park, College Park, in 1989. In 1990, he was an Associate Professor with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. Since 1995, he has been a Professor with the Department of Computer Science, National Chiao Tung University, where he is also the Director of Library. His current research interests include distance learning, internet technologies, distributed computing, cloud computing, and fog/edge computing.